



**Carnegie Mellon
Software Engineering Institute**

Fifth DoD Product Line Practice Workshop Report

John Bergey
Sholom Cohen
Matthew Fisher
Lawrence Jones
Linda Northrop
William O'Brien

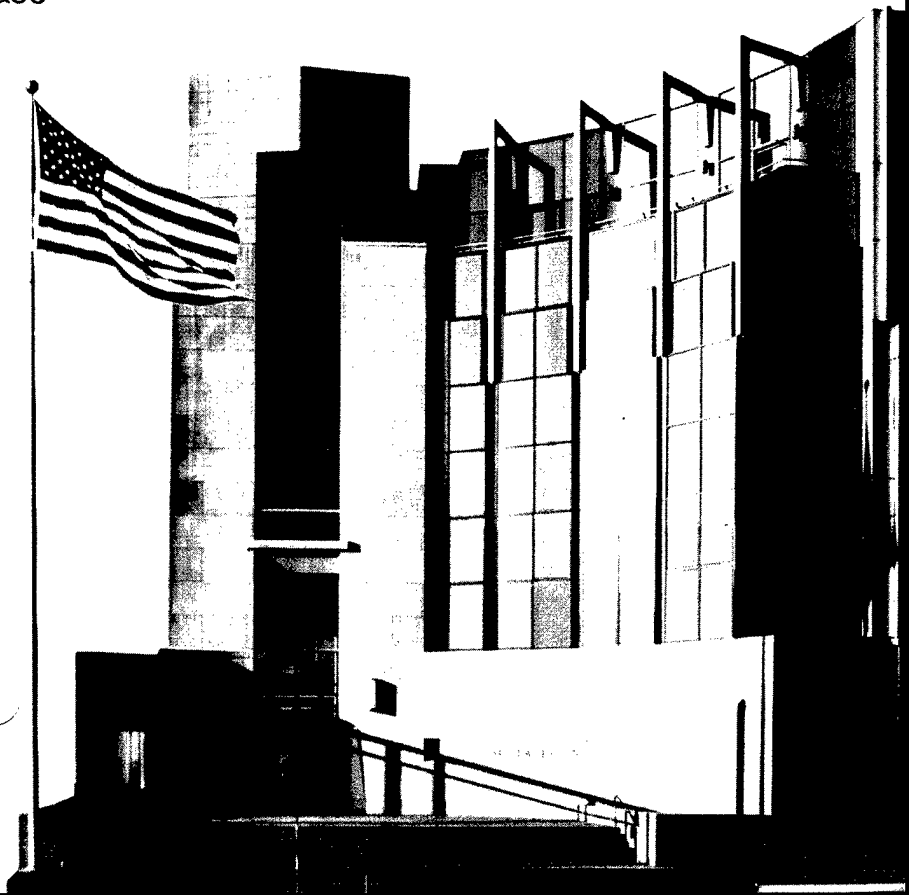
June 2003

DISTRIBUTION STATEMENT A

Approved for Public Release
Distribution Unlimited

TECHNICAL REPORT
CMU/SEI-2003-TR-007
ESC-TR-2003-007

20030822 126





**Carnegie Mellon
Software Engineering Institute**

Pittsburgh, PA 15213-3890

Fifth DoD Product Line Practice Workshop Report

CMU/SEI-2003-TR-007
ESC-TR-2003-007

John Bergey
Sholom Cohen
Matthew Fisher
Lawrence Jones
Linda Northrop
William O'Brien

June 2003

Product Line Practice Initiative

Unlimited distribution subject to the copyright.

This report was prepared for the

SEI Joint Program Office
HQ ESC/DIB
5 Eglin Street
Hanscom AFB, MA 01731-2116

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

FOR THE COMMANDER



Christos Scodras
Chief of Programs, XPK

This work is sponsored by the U.S. Department of Defense. The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

Copyright 2003 by Carnegie Mellon University.

NO WARRANTY

THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

Use of any trademarks in this report is not intended in any way to infringe on the rights of the trademark holder.

Internal use. Permission to reproduce this document and to prepare derivative works from this document for internal use is granted, provided the copyright and "No Warranty" statements are included with all reproductions and derivative works.

External use. Requests for permission to reproduce this document or prepare derivative works of this document for external and commercial use should be addressed to the SEI Licensing Agent.

This work was created in the performance of Federal Government Contract Number F19628-00-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.

For information about purchasing paper copies of SEI reports, please visit the publications portion of our Web site (<http://www.sei.cmu.edu/publications/pubweb.html>).

Table of Contents

1	Introduction	1
1.1	Why Product Line Practice?	1
1.2	About the Workshop	2
1.3	About This Report	4
2	State of Software Product Line Practice: A Digest of the SEI Overview Presentation.....	5
2.1	Introduction	5
2.2	Software Product Line Acquisition: A Companion to a Framework for Software Product Line Practice – Lawrence G. Jones, SEI	5
2.2.1	Product Line Context	5
2.2.2	The Framework for Software Product Line Practice	7
2.2.3	The Acquisition Companion to the Framework.....	10
3	DoD Software Product Line Experiences: Digest of DoD Presentations... 15	
3.1	Introduction	15
3.2	Product Lines for DoD Open Air Ranges – Edward P. Dunn, Naval Undersea Warfare Center	15
3.3	Acquisition of a Product Line for Army Live Training – Paul Watson, U.S. Army STRICOM	19
3.4	Development of a Product Line Architecture for Army Live Training – Wesley Milks, Lockheed Martin	21
4	Software Product Line Practices: Working Group Reports.....	25
4.1	Working Group 1 – Component Development and Software Systems Integration	25
4.1.1	Component Development	26
4.1.2	Software Systems Integration	27
4.2	Working Group 2 – Mining Existing Assets and Technical Planning	28
4.2.1	Mining Existing Assets	28
4.2.2	Technical Planning	29
5	Summary.....	33
Appendix	Workshop Participant Feedback on the Acquisition Companion	35

Bibliography	37
---------------------------	-----------

List of Figures

Figure 1: Essential Activities for Product Line Practice	7
--	---

List of Tables

Table 1:	Framework and Companion Practice Areas	12
Table 2:	Estimated Costs of Building Range Systems Without RangeWare.....	17
Table 3:	Estimated Costs of Building Range Systems with RangeWare.....	18

Abstract

The Software Engineering Institute (SEISM) held the Fifth Department of Defense (DoD) Product Line Practice Workshop in August 2002 in conjunction with the Second Software Product Line Conference (SPLC2). The workshop was a hands-on meeting to identify industry-wide best practices in software product lines; to share DoD product line practices, experiences, and issues; to discuss ways in which specific product line practices are accomplished within the DoD; and to obtain feedback on the Version 2 pre-release draft of *Software Product Line Acquisition: A Companion to a Framework for Software Product Line Practice* written by the SEI. This report synthesizes the workshop presentations and discussions.

1 Introduction

1.1 Why Product Line Practice?

An increasing number of organizations are realizing that they can no longer afford to develop multiple software products one product at a time: they are pressured to introduce new products and add functionality to existing ones at a rapid pace. They have explicit needs to achieve large-scale productivity gains, improve time to market, maintain a market presence, compensate for an inability to hire, leverage existing resources, and achieve mass customization. Many organizations are finding that the practice of building sets of related systems together can yield remarkable quantitative improvements in productivity, time to market, product quality, and customer satisfaction. They are adopting a product line approach for their software systems.

A software product line is a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way [Clements 02a].

Product line practice involves strategic, large-grained reuse as a business enabler. Some organizations have already experienced considerable savings by using a product line approach for software system production. Other organizations are attracted to the idea but are in varying stages of integrating product line practices.

In January 1997, the Software Engineering Institute (SEISM) launched the Product Line Practice Initiative to help facilitate and accelerate the transition to sound software engineering practices using a product line approach. The goal of this initiative is to provide organizations with an integrated business and technical approach to systematic reuse so they can produce and maintain similar systems of predictable quality more efficiently and at a lower cost.

The SEI has also refined the workshop results through work with collaboration partners, participation in other workshops, and continued research. This transition strategy has been executed in part by a series of product line workshops organized by the SEI.¹ Five of these workshops (in December 1996, November 1997, December 1998, December 1999, and December

SM SEI is a service mark of Carnegie Mellon University.

¹ The SEI also organized the first international Software Product Line Conference (SPLC1) held in Denver, Colorado, in August 2000 [Donohoe 00].

2000) brought together international groups of leading practitioners from industry to codify industry-wide best practices in product lines. The results of these workshops are documented in SEI reports [Bass 97, Bass 98, Bass 99, Bass 00, Clements 01]. These reports identify product line best practices, collectively refining and synthesizing some of the best ideas presented, and also identify issues that still require solution.

In March 1998, the SEI hosted its first Department of Defense (DoD) product line practice workshop, *Product Lines: Bridging the Gap—Commercial Success to DoD Practice*. Product line practices, DoD barriers and mitigation strategies, as well as similarities and differences between DoD product line practice and commercial product line practices were discussed and documented [Bergey 98]. A second product line practice workshop was held in March 1999. This workshop marked a turning point from the SEI perspective in that the DoD participants talked about how they were implementing or going to implement product lines, as opposed to the familiar lament from past DoD forums that it would be impossible to implement product lines within the DoD [Bergey 99a]. A third workshop was held in March 2000 [Bergey 00a] and a fourth workshop was held in March 2001 [Bergey 01]. At all four DoD workshops, the SEI was encouraged to continue to hold other DoD workshop events and to continue to bring best commercial practices to the DoD through these forums.

The SEI continues to refine the collective workshop results through work with collaboration partners, participation in other workshops, and continued research. In addition, the SEI is producing a conceptual framework for product line practice. The *Framework for Software Product Line Practice*SM written by the SEI (henceforth referred to as the framework) describes the foundational product line concepts and identifies the essential activities and practices that an organization must master before it can expect to field a product line of software or software-intensive systems successfully. The framework organizes product line practices into practice areas that are categorized according to software engineering, technical management, and organizational management. These categories represent disciplines rather than job titles. The framework is a living document that is evolving as experience with product line practice grows. Version 4.0 is described in the book, *Software Product Lines: Practices and Patterns* [Clements 02a]. Version 4.1 of the framework is available on the SEI Web site [Clements 02b].

1.2 About the Workshop

In conjunction with the Second Software Product Line Conference (SPLC2), the SEI held the fifth in the series of DoD Product Line Practice Workshops in August 2002 to achieve the following goals:

- Identify industry-wide best practices in software product lines.

SM Framework for Software Product Line Practice is a service mark of Carnegie Mellon University.

- Share DoD product line practices, experiences, and issues.
- Discuss ways in which specific product line practices get accomplished within the DoD.
- Provide feedback to the SEI on the *Software Product Line Acquisition Companion to a Framework for Software Product Line Practice* written by the SEI (henceforth referred to as the companion).

The workshop participants were referred to Version 3 of the framework and a pre-release draft of Version 2 of the companion to provide a common focus to structure the workshop presentations and discussions. All participants in this workshop were from the DoD acquisition and contractor community. They were invited based on our knowledge of their experience with and commitment to software product lines as either DoD system acquirers or DoD system contractors. Together we discussed the issues that form the backbone of this report.

The workshop participants included

- Robert Alexander, PM FBCB2, U.S. Army
- John Bergey, Product Line Systems Program, SEI
- David Bixler, TRW Tactical Systems
- W. Peter Blankenship, TRW
- Grady Campbell, Product Line Systems Program, SEI
- Joe Carlin, Argon
- Sholom Cohen, Product Line Systems Program, SEI
- Timothy Denmeade, Program Integration Directorate, SEI
- Patrick Donohoe, Product Line Systems Program, SEI
- Edward Dunn, Naval Undersea Warfare Center (NUWC), U.S. Navy
- Matthew Fisher, Product Line Systems Program, SEI
- Lawrence Jones, Product Line Systems Program, SEI
- Judy Kerner, The Aerospace Corporation
- Reed Little, Product Line Systems Program, SEI
- Wesley Milks, Lockheed Martin
- Alex Moy, PM TRCS-JTRS, U.S. Army
- Linda Northrop, Director, Product Line Systems Program, SEI
- Liam O'Brien, Product Line Systems Program, SEI
- Chuck Preh, PM TRCS, U.S. Army

- Dr. Rami Razouk, General Manager Computer Systems Division, The Aerospace Corporation
- Dennis Smith, Product Line Systems Program, SEI
- Daniel Stroka, PM FBCB2, U.S. Army
- Paul Watson, PM TRADE, U.S. Army
- Don Wilson, Raytheon Missile Systems

1.3 About This Report

This document summarizes the presentations and discussions at the workshop. While there is a brief overview of product line practice in Section 2, this report is written primarily for those in the DoD who are already familiar with product line concepts, especially those who are already working on or initiating product line practices in their own organizations. Acquisition managers and technical software managers should also benefit from this report. Those who desire further background information are referred to the following publications:

- *Basic Concepts of Product Line Practice for the DoD* [Bergey 00b]
- *A Framework for Software Product Line Practice, Version 4.1* [Clements 02b]
- *Software Product Line Acquisition: A Companion to a Framework for Software Product Line Practice, Version 2.0* [Bergey 03]
- *Software Product Lines: Practices and Patterns* [Clements 02a]

The remainder of this report is organized into four main sections that parallel the workshop format:

- Section 2: State of Software Product Line Practice, a digest of an SEI overview presentation
- Section 3: DoD Software Product Line Experiences, a digest of three DoD presentations
- Section 4: Software Product Line Practices: Working Group Reports, which details two working group reports that are organized around the suggested contents of the new practice areas that will be included in the next release (Version 3) of the companion. The emphasis and completeness of the information varies by group and by practice area.
- Section 5: Summary, which recaps the major themes of this report and suggests future directions

This report also contains an appendix that contains focused feedback on the pre-release draft of Version 2 of the companion.

2 State of Software Product Line Practice: A Digest of the SEI Overview Presentation

2.1 Introduction

Because this workshop was held in conjunction with SPLC2, attendees had the opportunity to attend the full-day product line tutorial, *Software Product Lines: Practices and Patterns*, making it unnecessary to provide the background presentations given at previous workshops. Larry Jones gave a brief overview addressing software product lines, the *Framework for Software Product Line Practice*, and the acquisition companion to the framework. For completeness, an expanded version of this information, adapted from the *Fourth DoD Product Line Practice Workshop Report*, is included in Sections 2.2.1 and 2.2.2 [Bergey 01].

2.2 Software Product Line Acquisition: A Companion to a Framework for Software Product Line Practice – Lawrence G. Jones, SEI

2.2.1 Product Line Context

2.2.1.1 What Is a Product Line?

An increasing number of organizations are realizing that they can no longer afford to develop multiple software products one product at a time: they are pressured to introduce new products and add functionality to existing ones at a rapid pace. They have explicit needs to achieve large-scale productivity gains, improve time to market, maintain a market presence, compensate for an inability to hire, leverage existing resources, and achieve mass customization. Many organizations are finding that the practice of building sets of related systems together can yield remarkable quantitative improvements in productivity, time to market, product quality, and customer satisfaction. They are adopting a product line approach for their software systems.

A software product line is a set of software-intensive systems sharing a common, managed set of features that satisfy the needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way [Clements 02a].

This definition is consistent with the definition traditionally given for any product line—a set of systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission. But it adds more; it puts constraints on the way the systems in a software product line are developed because substantial production economies can be achieved when the systems in a software product line are developed from a common set of assets in a prescribed way. The product line architecture and components are central to the set of core assets used to construct and evolve the products in a product line. This common, product line software architecture² capitalizes on commonalities in the implementation of the line of products and provides the structural robustness that makes the derivation of software products from software assets economically viable.

Each product in the product line is formed by taking applicable components from the base of common assets, tailoring them as necessary through preplanned variation mechanisms such as parameterization or inheritance, adding any new components that may be necessary, and assembling the collection according to the rules of the product line architecture. Building a new product (system) becomes more a matter of assembly or generation than creation. For each software product line there is a predefined guide or plan that specifies the exact product-building approach.

Product line practice involves strategic, large-grained reuse as a business enabler. The key concepts are

- **the use of a common asset base** (with the architecture being the pivotal asset)
- **in the production** (according to a predefined and documented production plan)
- **of a set of related products** (whose scope has been clearly defined and validated with a business case)

2.2.1.2 The State of Product Line Practice

A number of organizations have achieved their product line goals. They have already gained order-of-magnitude improvements in efficiency, productivity, and quality through the strategic software reuse afforded by a product line approach. However, even more important than significant cost savings, product line practice enables an organization to get its products to market or field at the right time. Time has emerged as a critical success factor in a number of highly competitive product lines, such as cellular phones, pagers, and printers. If a product reaches the marketplace several months after its competitor, it may have lost its window of opportunity and become a failure regardless of its features or cost. To read about some of the many success stories, see the previously referenced workshop reports, the case studies written by Clements and Northrop [Clements 02a], and the proceedings of SPLC2 [Chastek 02].

² A software architecture of a computing system is the structure or structures of the system that consist of software elements, the externally visible properties of those elements, and the relationships among them [Bass 03].

Many more organizations are now attracted to the concept of software product lines to address their needs for faster, better, and cheaper software production. Before moving to a product line approach for software, an organization should first identify its business goals and then determine if product line practice is a viable strategy for reaching those goals. Software product line practice is not a panacea, but it has demonstrated significant advantages in many organizations that had a business case to support product line practice.

2.2.2 The Framework for Software Product Line Practice

Despite the differences among organizations involved in product line efforts, there are essential activities and practices common to all successful product lines. We describe the essential activities and practices in the framework (which is conceptual), available as a Web-based, evolving document and targeted primarily at members of organizations who are in a position to make or influence decisions regarding the adoption of product line practices.³

As depicted in Figure 1, at its essence, fielding a software product line involves *core asset development*, product development from the core assets, and management to staff, orchestrate, and coordinate the entire product line effort. The arrows signify the high degree of iteration involved and the fact that there is no prescribed order as to how these activities take place.

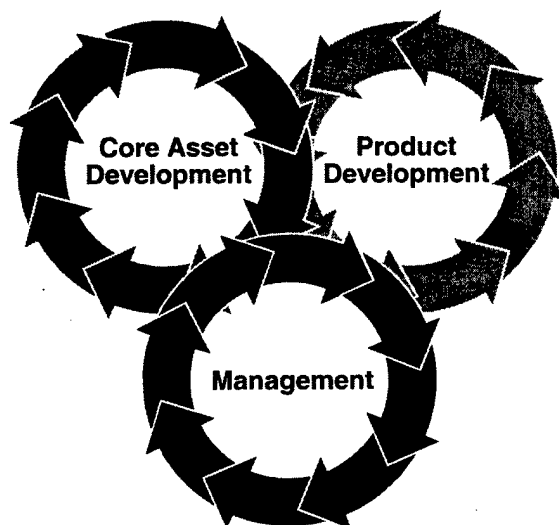


Figure 1: Essential Activities for Product Line Practice

³ At the time of this workshop, Version 3.0 of the framework was available on the SEI Web site. Workshop participants were asked to read that version.

On the left side of the figure, the critical core assets involved are the architecture and components. Inputs to the development and acquisition of core assets are product constraints found by analyzing the similarities and differences of current and projected products; production constraints such as might be found in a technical architecture; a production strategy for the assets; an inventory of preexisting assets; and styles, patterns, and architectural frameworks. The outputs are the core assets, a preliminary list of the products they will support, and a production plan for how the core assets will be used in the development or acquisition of products.

On the right side of the figure, individual products are developed or acquired from the core assets using the production plan that has been established. Product requirements are developed and refined with the existing core assets in mind, and products that systematically reuse the core assets are output.

There is a strong feedback loop between the core assets and products. Core assets are refreshed as new products are developed. In addition, the value of the core assets is realized through the products that are developed from them. As a result, the core assets are made more generic by considering potential new products on the horizon. There is a constant need for strong and visionary management to invest the resources in the development of the core assets and to develop the cultural change required to view new products through the filter of the core assets.

There are essential practices in a number of specific areas that are required to produce the core assets and products in a product line and to manage the process at multiple levels. The framework describes the essential practice areas for software engineering, technical management, and organizational management, where these categories represent disciplines rather than job titles. For individual practice areas, the framework provides

- an introductory description of the practice area
- aspects of the practice area that are peculiar to product lines
- how the practice area is applied to core asset development
- how the practice area is applied to product development
- specific practices in the practice area
- risks in the practice area
- additional references

2.2.2.1 Software Engineering

The software engineering practice areas include

- Architecture Definition
- Architecture Evaluation
- Component Development
- COTS Utilization
- Mining Existing Assets
- Requirements Engineering
- Software System Integration
- Testing
- Understanding Relevant Domains

2.2.2.2 Technical Management

The technical management practice areas include

- Configuration Management
- Data Collection, Metrics, and Tracking
- Make/Buy/Mine/Commission Analysis
- Process Definition
- Scoping
- Technical Planning
- Technical Risk Management
- Tool Support

2.2.2.3 Organizational Management

Organizational management is the name we give to the management of the business issues that are visible at the enterprise level, as opposed to those at the project level. Enterprise management includes those practice areas necessary to position the enterprise to take full advantage of the product line capability. The organizational management practices include

- Building a Business Case
- Customer Interface Management
- Developing an Acquisition Strategy
- Funding
- Launching and Institutionalizing
- Market Analysis

- Operations
- Organizational Planning
- Organizational Risk Management
- Structuring the Organization
- Technology Forecasting
- Training

2.2.2.4 Future Direction of the Framework

The framework is intended to be a living document. We made a decision to make it available before all of the practice areas were complete. Version 1.0 was the first step in engaging the community to provide feedback on the framework's accuracy and usefulness. We incorporated community feedback and our growing experience base in Version 2.0 and included a frequently asked questions section. Version 3.0 achieved further stability and maturity, leading to Version 4.0 [Clements 02a] and then modestly updated in Version 4.1 [Clements 02b]. We are encouraged that more than 40 organizations have reported to us their use of the framework. Future versions of the framework will build on the current foundation and continue to incorporate feedback and our experience.

2.2.3 The Acquisition Companion to the Framework

Originally, the framework was envisioned as meeting the needs of two broad communities interested in implementing a software product line approach: software development organizations and DoD acquisition organizations. However, these two communities differ in at least one obvious and significant way. While software development organizations will generally use acquisition in a supplemental role, DoD acquirers rely primarily (if not exclusively) on acquisition to obtain systems. The framework authors discovered that trying to address both communities in a single document proved to be cumbersome, often resulting in awkward wording and compromising the needs of both groups.

Our solution to this dilemma was to focus the framework on the needs of software developers and create a second document, *Software Product Line Acquisition: A Companion to A Framework for Software Product Line Practice*, to address the needs of DoD acquirers.

We assume that the audience for the companion consists of experienced, responsible acquirers. Thus, the companion is not a primer on basic acquisition. It will not appeal to the acquirer who wishes merely to "outsource and forget." While we have targeted the DoD, our initial experience shows that the companion is also relevant to other government organizations.

Our development strategy is similar to that used for the framework. We are developing the companion iteratively, releasing increasingly polished and complete versions over time. The first version, released in July 2001, contained roughly one-third of the envisioned content. The second version contains roughly two-thirds of the envisioned content. Participants in this workshop received a pre-release draft of Version 2 of the companion. The feedback received during this workshop resulted in changes to Version 2 and provided grist for Version 3.

Version 2 of the companion is structured into three sections. Section 1 provides a general introduction to the document. Section 2 describes software product line concepts, briefly addresses the general DoD acquisition context, and concludes with a broad overview of major concepts and issues in software product line acquisition. The heart of the document, Section 3, is structured using the same 29 practice areas and 3 practice area categories found in the framework.

Consistent with practice areas in the framework, a companion practice area describes a body of work or collection of activities that acquirers must master to acquire software product lines successfully. The actual product line will be developed by a second organization, called the supplier, to which the framework applies. The practice areas in both documents are classified according to one of three categories: software engineering, technical management, and organizational management. The complete list of practice areas by category is shown in Table 1.

Software Engineering Practice Areas	Technical Management Practice Areas	Organizational Management Practice Areas
Architecture Definition Architecture Evaluation Component Development* COTS Utilization Mining Existing Assets* Requirements Engineering Software System Integration Testing* Understanding Relevant Domains	Configuration Management* Data Collection, Metrics, and Tracking Make/Buy/Mine/Commission Analysis Process Definition Scoping Technical Planning* Technical Risk Management Tool Support	Building a Business Case Customer Interface Management Developing an Acquisition Strategy Funding Launching and Institutionalizing Market Analysis Operations Organizational Planning Organizational Risk Management Structuring the Organization Technology Forecasting* Training*

** Indicates practice areas to be added to Version 3.0 of the companion*

Table 1: Framework and Companion Practice Areas

Software engineering practices are those practices necessary to apply the appropriate technology to create and evolve both core assets and products. In the typical case, the details of carrying out these practices are largely the concern of the supplier. The acquirer's role in these practices is generally to

- set direction for the supplier and establish the acquisition team's role through the appropriate acquisition documents
- monitor and review the supplier's progress
- monitor the supplier's technical process and products
- participate (strongly) in the establishment and evolution of requirements

Technical management practices are those practices necessary to manage the creation and evolution of the core assets and the products. Again the details of carrying out these practices are typically the concern of the supplier. The acquirer's role in technical management practices is generally to

- set direction for the supplier and establish the acquisition team's role through the appropriate acquisition documents

- monitor and review the supplier's progress
- monitor the supplier's technical management activities
- participate (strongly) in product line scoping

Organizational management practices are those practices necessary for the orchestration of the entire product line effort. Here we have a shift in relative emphasis from the supplier to the acquirer. Generally, the acquirer will have the major responsibilities in these practices. While the supplier might have parallel activities in some practices (e.g., "Organizational Risk Management" or "Organizational Planning"), in other practices the supplier might have little or no role (e.g., "Building a Business Case" or "Developing an Acquisition Strategy").

The bulk of Section 3 consists of detailed information about the practice areas. Each practice area is presented with the following information:

- summary of framework information – a synopsis of the practice area description from the framework (typically from a supplier's point of view).
- aspects of the practice area that relate to acquisition – a description of what makes product line acquisition different from acquiring a single system and special concerns in product line acquisition.
- frequently asked questions
- references

In keeping with our philosophy of building a living document in an iterative fashion, subsequent versions might depart from this structure. As mentioned earlier, we have already incorporated feedback from this workshop and other sources prior to publishing Version 3.0. We invite readers to provide feedback and suggestions for improving the document.

3 DoD Software Product Line Experiences: Digest of DoD Presentations

3.1 Introduction

The following three presentations related to DoD software product lines provided the DoD context that was helpful in framing subsequent discussions:

1. "Product Lines for DoD Open Air Ranges" by Edward P. Dunn
2. "Acquisition of a Product Line for Army Live Training" by Paul Watson
3. "Development of a Product Line Architecture for Army Live Training" by Wesley Milks

The second and third presentations address the same system but from the perspective of the acquirer and the supplier, respectively.

3.2 Product Lines for DoD Open Air Ranges⁴ – Edward P. Dunn, Naval Undersea Warfare Center

The Engineering, Test, and Evaluation Department of the Naval Undersea Warfare Center (NUWC) – Division Newport has developed a software product line asset base, named RangeWare, to support test range operations, tracking, archiving, playback, debriefing, analysis, and display applications. NUWC's current and future customers for the product line include major test and evaluation (T&E)/training ranges, specialized test facilities, subsystems, and associated simulations. NUWC has fielded a product line of range systems for current customers using the asset base. After several pilot applications of RangeWare, NUWC is now taking RangeWare into a sustainment phase, expanding the coverage of the asset base in terms of object and distribution services, as well as applying the assets to new systems.

Dunn's presentation on RangeWare focused on the scope, goals, and architecture of the product line effort. He also described the projects currently underway that are using RangeWare and the product line practice areas that are applied at NUWC. Dunn characterized the product

⁴ More details can be found in a joint NUWC-SEI technical report written by Cohen, Dunn, and Soule [Cohen 02].

line as systems that include a large number of sensors requiring intensive, real-time processing. Range systems are often safety critical and require multiple real-time displays. The presentation highlighted examples where RangeWare has been used and discussed its object-oriented, platform-independent, highly adaptable software architecture.

NUWC set several goals that RangeWare has achieved:

- support current range capabilities
- position ranges for innovative technology
- deliver systems with
 - higher quality
 - shorter development time
 - lower total ownership cost

RangeWare assets include components for implementation and integration of range capabilities. An application programming interface (API) and API implementation allow easy creation and manipulation of RangeWare objects, including sensors, displays, and controls. RangeWare applications, such as data viewers and recorders, use or implement RangeWare objects. Architectural framework assets support the building of common types of applications. Assets also include data interfaces to non-RangeWare systems.

In developing RangeWare, NUWC applied many of the framework practice areas. Dunn highlighted key practices and challenges in the following areas:

- “Software System Integration” – RangeWare includes a master repository of assets and scripts for composition of assets. RangeWare uses a mix of government and commercial off-the-shelf software. Developers have capitalized on cross-system commonality for system integration. This commonality includes specialized hardware interfaces, displays, and mission capabilities. The integration approach supports a high degree of runtime flexibility based on interfaces to other systems, both those that use RangeWare and those that don’t.
- “Data Collection” – NUWC has used history information plus data collected from current programs as input to a Software Process Improvement Initiative (SPII). NUWC’s historic data is limited to large-grained cost data.⁵ However, these data demonstrate that projects have mastered opportunistic reuse. NUWC is currently collecting more precise cost data and line of code counts. The SPII will identify the metrics that must be measured and establish collection mechanisms for the required data.
- “Configuration Management” (and quality assurance) – NUWC has installed a formal configuration management (CM) process and applies formal quality assurance (QA) for

⁵ The data is available only at the project level and is based on total lines of code.

system builds. A Software Configuration Change Board authorizes and tracks change proposals as well as software modifications. The CM system also informs stakeholders (those affected by modified software). The current QA system is somewhat limited. Because funding of RangeWare is limited, any regression testing is limited to situations where a RangeWare application is used in an actual system build. NUWC does not “sponsor” QA for individual programs using the assets.

- **Building a Business Case** – NUWC has assembled cost data that show savings of between \$3 and \$5 million on development costs and over \$15 million on maintenance costs. The business case takes the form of a succession of models: an intuitive model based on a sample of typical NUWC systems, a predictive model based on actual systems that NUWC plans to build, and an experiential model that uses data from actual systems that used RangeWare. The actual cost data strengthens and refines the business case with greater dependability in the predictive model. Table 1 shows estimated costs of building a set of ranges without RangeWare. Table 2 estimates the costs using RangeWare assets to support building the same set of ranges.

Program Name	Lines of Code (in 000s)	Development Cost (in 000s)	Maintenance Cost per year (in 000s)	Number of Years in Maintenance	Cost with Maintenance (in 000s)
Major Range A	700	\$7,000	\$700	20	\$21000
Small Range B	400	\$4,000	\$400	20	\$12000
Subsystem A	250	\$2,500	\$250	10	\$5000
Subsystem B	150	\$1,500	\$150	10	\$3000
Subsystem C	50	\$500	\$50	10	\$1000
Eng Prot A	150	\$1,500	\$150	4	\$2100
Eng Prot B	50	\$500	\$50	1	\$550
Total		\$17,500			\$44,650

Table 2: *Estimated Costs of Building Range Systems Without RangeWare*

Program Name	Lines of Code (in 000s)	Development Cost (in 000s)	Maintenance Cost per year (in 000s)	Number of Years in Maintenance	Cost with Maintenance (in 000s)
Major Range A	210	\$4375	\$437.50	20	\$13125
Small Range B	120	\$2500	\$250.00	20	\$7500
Subsystem A	75	\$1562	\$156.25	10	\$3125
Subsystem B	45	\$937	\$93.75	10	\$1875
Subsystem C	15	\$312	\$31.25	10	\$625
Eng Prot A	45	\$937	\$93.75	4	\$1312
Eng Prot B	15	\$312	\$31.25	1	\$343
Asset Base	150	\$2250	\$225.00	20	\$6750
Total		\$13,185			\$34,556

Table 3: Estimated Costs of Building Range Systems with RangeWare

- “Structuring the Organization” – Dunn discussed the interactions between the asset group maintaining RangeWare and the programs using assets to build products. NUWC works with ranges (NUWC customers) to determine their requirements and the suitability of using RangeWare as the basis for system development. Team leaders located at NUWC are responsible for cost, schedule, and performance in developing these systems for range customers. In addition to RangeWare suitability, the acceptance of RangeWare by team leaders is also necessary. One of the challenges facing NUWC is strengthening the resolve of team leaders and their continued use of RangeWare. NUWC must also find ways to resolve funding issues and improve management of the interface with the customer.
- “Operations” – The RangeWare group works with customers to determine whether a program’s requirements align with RangeWare capabilities. Actual product development follows the Product Builder Pattern [Clements 02a]. NUWC first determines if the customer’s requirements are within the scope. Questions NUWC addresses in making the determination include changes to existing assets, the need for new RangeWare assets, opportunity for shared developments, system-unique requirements, and the overall appropriateness of RangeWare as a solution. NUWC manages other assets that might be more suitable in specific cases depending on factors such as the expected life of the new system.

Dunn emphasized the importance of achieving immediate benefits while planning for the broader implementation of product line goals. Organizations should build on existing relationships for their initial customer base while identifying new markets. The organization must have clear business goals and known architectural drivers to maintain the integrity of the product line. Without them, the product line might in short order dissolve once again into in-

dividual projects. Selling products requires more than an architecture or “cool” technology. The product line organization must be able to demonstrate real benefits and field a team that can both influence and bootstrap capabilities in user organizations.

3.3 Acquisition of a Product Line for Army Live Training – Paul Watson, U.S. Army STRICOM

The Department of the Army has a need to modernize its Maneuver Combat Training Centers in addition to providing live environment training at various Homestations, Live Fire ranges and Military Operations in Urban Terrain (MOUT) sites. Current systems satisfying this need are aging and typically were acquired as point solutions to meet the requirements. Part of this aging is reflected in instrumentation systems that are based on older technology and don't take advantage of the newer technologies available. Many of these solutions do not interoperate, a critical concern. Many of the near-term problems at the Combat Training Centers (CTCs) are urgent. The CTC Council of Colonels has validated near-term requirements for fixes to some of the problems, and these fixes have been prioritized. Most of these efforts, however, are funded inadequately or not at all.

The U.S. Army's Project Manager for Training Devices (PM TRADE) has looked across the various training systems and their associated requirements in an effort to identify commonality, using formal domain analysis techniques. The “look” is with the vision of employing strategic reuse concepts, thereby improving the quality of the training while significantly reducing the high logistics, training, and maintenance costs associated with the traditional live training alternatives. Taking advantage of this commonality (in a strategic reuse or product line sense) would reap an economic benefit both in time and resources. Again, in this look, the PM TRADE wants to ensure the evolving training systems will be interoperable. An examination of the various user requirements indicated that at the Operational Requirements Document (ORD) level there is 90% commonality embodied in the training system requirements. But that does not translate into 90% cost savings, as the 10% of variability within ORD requirements can translate to higher percentages of system costs. Still, the domain analysis has clearly indicated that there is a viable business case for pursuing strategic reuse within the family of live training systems the PM TRADE must field. The PM has formalized this strategic approach within the overarching live training transformation (LTT) product line acquisition strategy.

The vision of the LTT modernization effort is to

- provide effective and affordable training everywhere live training occurs
- apply modern, modular, and reconfigurable technology
- ensure C4I interoperability

- ensure live/virtual/constructive (L/V/C) interoperability

The proposed solution is to take a product line approach in

- developing a common training instrumentation architecture
- employing common, reusable, adaptable hardware and software
- ensuring rapid integration throughout the force
- having an embedded capability

The LTT product line encompasses at least eight separately funded major programs, some with multiple fielding locations. The first and most critical program involves the creation of the LTT product line architecture, referred to as the Common Training Instrumentation Architecture (CTIA). The first mature delivery of the CTIA, Version 1.0, was completed in September 2002. Future versions will evolve using spiral development techniques to keep pace with advancements in technology and to adapt to new requirements incorporated into the product line. The other programs under the LTT umbrella, referred to as the “products,” include complete instrumented range solutions for the CTCs, Homestations, Live Fire ranges and MOUT sites.

The development life cycle for LTT products extends to fiscal year 2012, with a use and support life cycle extending beyond fiscal year 2020. The overarching LTT product line Single Acquisition Management Plan (SAMP) formally identifies the relationship between the CTIA and the products within the family. The PM successfully defended this strategy within the Army’s user and resource communities, and established a new and separate funding line for the CTIA. The CTIA program follows a typical phased acquisition path of definition, design development, and deployment, under continued management. The initial phase, conducted between fiscal year 1999 and 2001, involved the definition of the CTIA as documented in the first release of the architecture referred to as V0.1. The last phase discussed, referred to as LTT product line management, showed a continual improvement in the LTT architecture and components with plans to embrace future combat systems and operations.

The CTIA integrated product team (IPT) used domain analysis and the input from numerous users to define the CTIA. This definition addressed

- reusable components and repository
- leveraging C4I components
- JTA-A, DII-COE and HLA compliancy
- ATIA-compliant data interchange
- links to L/V/C simulations

- iterative, spiral development processes
- essential products for NTC-OIS and all other LTT deliveries
- prototypes of high-risk capabilities

The CTIA is an overarching architecture encompassing both virtual (sensors) and constructive (simulation/stimulation) elements. At the same time, the architecture must address the needs of the various organizations that use such training systems: institutions, Homestations, CTCs, and deployed units.

As an acquisition program, PM TRADE is following the tenants of the DoD 5000 framework. This means that approval for the program must be obtained along with fiscal and personnel resources to carry out the acquisition. The program office activities and subsequent contractual requirements also follow a tailored version of the processes from the Software Productivity Consortium's *Reuse-Driven Software Processes Guidebook* [SPC 93].

PM TRADE received approval to proceed and upfront funding to define the architecture and develop core assets. They also are receiving funding to sustain the product line approach although there are concerns that available funds for live training investment will be diverted to near-term fixes of identified problems with current implementations, and away from long-term product line solutions (not an unusual situation). Both efforts need to be supported. But diverting resources from the live training environment effort will prevent PM TRADE from getting ahead of the bow wave of force modernization (digitization, transformation, emerging weapons platforms, and changing operational environments).

A key lesson learned is that there is a significant advantage to having an acquisition manager with the desire, funding, responsibility, and authority to implement product lines. This is in contrast to having technical people struggle to convince the organization of the benefits of product lines within the context of its mission. This underscores the fact that the decision to employ a product line approach is a business decision.

3.4 Development of a Product Line Architecture for Army Live Training – Wesley Milks, Lockheed Martin

The Department of the Army has a need to modernize its Maneuver CTCs in addition to providing training systems at various Homestations, Live Fire ranges, Digital Multi-Purpose Range Complexes, and MOUT sites. An initial domain analysis indicated that there is sufficient commonality among the training sites to apply domain engineering principles and processes during the development of the training systems applications. This emphasis on commonality is expected to improve the quality of training while significantly reducing the high

logistics, training, and maintenance costs associated with live training. However, there is considerable variability that needs to be incorporated and managed.

The program strategy is to first develop and baseline a CTIA. The CTIA is being developed in an evolutionary fashion with an initial set of tasks that include

- production of a product line architecture specification and a product line architecture framework that describes the common operational requirements of the Army's Maneuver CTCs
- analysis, design, development, integration, testing, simulation, and verification of a subset of CTIA-compliant LTT common, reusable components
- analysis, design, development, integration, testing, simulation, and verification of the system and subsystem prototypes of Objective Instrumentation System capabilities
- a synchronization process among the family of products that will leverage product line deployment products and lessons learned with existing common components and products

The CTIA objectives are to conceptualize, design, develop, produce, modify, test, deliver, and sustain a component-based product line architecture for instrumentation systems.

The domain engineering and product line development process has been tailored from the *Reuse-Driven Software Processes Guidebook* [SPC 93]. From the *Guidebook*, certain tasks were selected and adapted to specific areas within the CTIA and some tasks were excluded. The goal of the tailoring was to generate a phased domain engineering approach that allowed logical progression of domain definition, domain analysis, architecture specification, architecture validation, and product line development.

The CTIA effort was characterized as a contractor-developed architecture to support a government-owned business case. The characteristics of the CTIA align with the leveraged synthesis category as defined in the *Guidebook*, and that section of the *Guidebook* was used as the starting point for tailoring.

The CTIA team has completed the first pass of all the domain engineering tasks and is currently applying the product line deployment process on two systems.

The CTIA effort started with a small budget and a small team. The use of a small team turned out well as there was a need to make changes regularly, which may have been more difficult with a larger team. More people were added to the team as work progressed. Components were developed over three six-month development spirals. The development team consists of two dozen developers and the first spiral is approximately 100 KLOC with the development

taking place in a Linux environment. The main quality attributes of the system being developed are scalability (being able to support a large number of players) and modifiability (as technology changes and new technology is inserted).

The CTIA is a component-based architecture and as such impacts more than just the LTT development and acquisition effort. The CTIA represents a set of business objectives that define the development process, future investment, life-cycle sustainment approaches and customer priorities. The evolution of the CTIA has its basis in business objectives and the CTIA will continue to evolve as those business objectives are further defined.

4 Software Product Line Practices: Working Group Reports

Following the plenary presentations, workshop participants divided themselves into two working groups. Each working group discussed the acquisition companion in general⁶ and performed a more detailed analysis of two practice areas from the seven to be added to Version 3 of the companion. The seven new practice areas that will be included in Version 3 of the companion are⁷

- software engineering practices
 - Component Development*
 - Mining Existing Assets*
 - Software System Integration*
- technical management practices
 - Configuration Management
 - Technical Planning*
- organizational management practices
 - Technology Forecasting
 - Training

The style and format of reporting differed between the groups. The following sections contain summaries of the work of the two groups.

4.1 Working Group 1 – Component Development and Software Systems Integration

This group took the approach of identifying important acquisition issues related to the practice area and then discussing them. They were sometimes able to articulate important questions (a valuable contribution in itself) but not always able to provide a definitive answer.

⁶ General feedback on the framework and the companion is contained in the appendix.

⁷ The four practice areas chosen by the groups are indicated with an asterisk (*).

4.1.1 Component Development

4.1.1.1 Description

As a starting point, the group was provided with the following digest of the framework contents for this practice area.

The term "component" is generic. For purposes of product lines, components mean the units of software that go together to form whole systems (products), as dictated by the software architecture for the products and for the product line as a whole. The practice of component development refers to the production of components that implement specific functionality within the context of an existing architecture. The functionality is encapsulated and packaged, an interface is presented and it is integrated with other components using an interconnection method. Successful component development for a product line requires a development process that will produce high-quality (trustworthy) and well-documented components that can be quickly located in an asset repository, with assurance that they will meet the pending need, and can be easily varied, according to the production plan, for use in the different contexts implied by the various applications in which they will be used.

Component development is the implementation heart of a product line. Without components that are high quality, application builders will be unwilling to commit to using the core asset base and the product line approach will founder. Components that are insufficiently documented to support either users or implementers will result in reliance upon individuals to answer key technical questions, and individuals in the core asset effort are in critically short supply and subject to untimely departure.

4.1.1.2 Group Discussion Points

The group discussed the following points.

- Acquisition of software components is dependent on the software architecture. A well-defined software architecture is crucial before component acquisition can take place.
- For an acquirer, it is important to identify different categories of components, characterized by their importance. The more important categories need more careful acquisition oversight. The relative importance of components will depend on the software architecture. Examples of important component categories include
 - infrastructure components; for example: widely used services, communications mechanisms, middleware, or an implementation of an API
 - tools necessary to use the architecture effectively
- Some components may be very large grained. It is important for the acquirer to have insight into the size and complexity of a component to manage its acquisition.
- The acquirer needs to see that the right developer is chosen to develop the right component. If component developers are subcontractors, the acquirer needs to have appropriate

visibility into the prime's selection process for subcontractors to see that this is done properly. Another (non-recommended) alternative is for the government to contract for the development of the components, thus serving in an integrator's role. For a typically staffed acquisition office, the group felt this would lead to an acquisition nightmare, particularly if many components were involved.

- Different acquisition strategies may be appropriate for components that are intended for inclusion in the reusable core assets versus those that will be product specific. The acquirer should consider whether any product-specific components might be candidates for incorporation into the core assets and how this will be accomplished.

4.1.2 Software Systems Integration

4.1.2.1 Description

Again, the group was provided with a summary of the framework contents for this practice area as a starting point.

Software systems integration refers to the practice of combining individual software components into an integrated whole. Software is integrated when components are combined into subsystems or when subsystems are combined into products. Integration of components is performed according to a production plan. Components should be relatively large with well-defined and well-documented interfaces.

In a product line effort, software system integration occurs during the installation of core assets into the asset base and also during the building of an individual product. Software systems integration relies on core assets that have been developed, acquired, or mined and a production plan. Early emphasis is placed on planning for integration by establishing component interfaces and creating the production plan. As systems are developed, the architecture is validated, the production plan is tested, and integration becomes more straightforward and predictable. Application developers should become more comfortable with components provided by the asset development group because their interfaces have been defined, they have been tested, and components work with one another. As a result, the cost and time of integration is significantly reduced when products are developed.

4.1.2.2 Group Discussion Points

The group discussed the following points:

- First, decide on an integration model. Who will do the integration? A contractor? The government? When and where the integration will occur is also important. With modern runtime binding (desirable for "plug and fight" use), will the end user be an "integrator" in some sense? If so, safeguards are needed.

- As an acquirer you need to have assurance that the integrated system conforms to the constraints of the architecture. You need to see that this happens in an architecture-based development lifecycle. Interfaces need to be checked for conformance prior to integration.
- The acquirer should require an integration plan from the supplier to allow insight into this phase.
- Product descriptions are needed to specify such things as version configuration requirements and associated rationale.

4.2 Working Group 2 – Mining Existing Assets and Technical Planning

This group's discussions followed the practice area structure. After considering the framework summary, each practice area was discussed in terms of aspects related to product line acquisition and frequently asked questions. Perhaps not surprisingly, this group also noted that identifying important issues and questions is easier than providing definitive answers.

4.2.1 Mining Existing Assets

4.2.1.1 Description

The group was provided with the following summary of the framework contents for this practice area.

The term "mining existing assets" simply refers to finding and rehabilitating useful legacy design or code from an organization's existing systems' portfolio, and reusing it within a new application. Software product line development usually begins with an analysis of related legacy systems. The intellectual capital of an organization is often tied to the design, to the code, or to the algorithms resident in its legacy systems. It is therefore important to inventory existing assets and to obtain an understanding of both their current functional and architectural features. Strategies must be developed for mining existing assets that are desirable to include in the core asset base. The strategies must render the assets compatible with the product line architecture and robust enough to be used across the family of products in the product line. Assets that require extensive renovation and rehabilitation will likely not be cost-effective to mine.

4.2.1.2 Aspects Related to Product Line Acquisition

The group discussed the following points:

- Data rights and intellectual property issues have to be addressed.

- There may be security issues for mined assets, for example
 - What are the implications for accrediting a mined asset?
 - The source of the asset is important for accreditation. For example, you cannot get accreditation for open source software or software produced by a foreign company.
 - A system security accreditation agreement (SSAA) needs to be written for products. There is not typically an umbrella agreement for a product line.
 - Can a mined asset be incorporated as a component and still meet the SSAA?
- General considerations regarding asset mining are
 - requirements for asset documentation
 - requirements specification for the assets
 - who owns the asset and who does the mining?
- Don't forget to consider mining government-owned assets other than code, such as
 - requirements
 - design
 - documentation
 - test cases
 - models and algorithms

4.2.1.3 FAQs

The group suggested including the following frequently asked questions in the companion.

- Does a product line architecture have to be in place before mining activities start?
- Does it matter whether you are mining for a core asset or a product-specific asset?
- What are the ownership issues for a mined asset?
- What are the considerations if COTS is embedded in code that is a candidate for mining?

4.2.2 Technical Planning

4.2.2.1 Description

The group was provided with the following summary of the framework contents for this practice area.

Technical planning involves software product line planning at the project level. By project we mean an undertaking typically requiring concerted effort that is focused on developing or maintaining a specific product or products. Typically a project has its own funding, accounting, and delivery schedule. In a product line context, a project might be responsible for developing specific core assets or for developing a specific product from the core assets.

A related practice area is "Organizational Planning," which focuses on the strategic planning that transcends projects. The process for generating plans is often very similar regardless of the organizational level at which it is applied. The process should differ primarily by who is involved and the scale of the effort to be planned. These two practice areas will also differ by the types of plans developed and their scope.

4.2.2.2 Aspects Related to Product Line Acquisition

The group discussed the following challenges in planning a product line approach:

- Unless there is some sort of umbrella contract in place, each acquisition must be conducted separately. This could be tricky.
- You must determine the need for and plan for the acquisition of product-unique assets.
- A product line acquisition will often have critical dependencies on other acquisitions such as
 - funding
 - priorities
 - schedule
- Schedule issues are particularly problematic.
 - There may be intra-project as well as inter-project dependencies.
 - If you don't own (or have responsibility for) the core assets, your critical path may be outside of your control.
 - A balance of priorities between core assets and products is necessary.
- There are liability issues. If an asset is faulty, who is accountable?

The group also provided a lot of good advice for product line planning.

- Know the different audiences for your plans and consider different versions for each audience. For example, in product-oriented plans there is a need to address both internal and external audiences. You may need to shield the product line details from a customer. As another example, technical plans must clearly delineate the perspective of the supplier and acquirer. Finally, technical plans can serve a valuable communication role to new team members to help them understand the product line context.
- There needs to be easy mechanisms to tie the product line strategic plans to the technical plans and convey the essential strategic information to technical planners. Similarly, people who are doing technical planning need to have procedures to raise warnings to strategic planners. Lack of such procedures is a common failure. Technical planners may uncover problems that are not apparent in strategic plans. It is important to manage expectations and for both sides to be honest about whether details of an approach will work.

- You should be prepared to assist related programs that are not formally part of the product line effort. You need to account for this in your plans.
- Technical plans for software product lines should address hardware dependencies. You should require the developer to make these specifications clear.
- Because planning is not a one-time activity, you should have a product line planning forum. This forum should meet regularly to review progress and determine if changes to the strategic and technical plans are necessary.
- Product line IPTs are useful for planning.
 - IPTs should have broad representation.
 - IPTs can work to resolve priorities on core asset needs.
 - IPT members require education on product line principles to be fully effective.
 - Consider the incentives for different members and attempt to plan for objectives that support those incentives.

4.2.2.3 FAQs

The group suggested including the following frequently asked questions in the companion.

- How do you adjust your planning when an external organization chooses not to use a standard development and test environment?
- How does the framework (companion) address the Lead System Integrator (LSI) approach?

5 Summary

The SEI's Fifth DoD Product Line Practice Workshop explored the product line practices of organizations in the DoD community in light of best commercial practices and government experience in software product lines. The presentations and discussions again validated the pivotal pieces of the SEI's *Framework for Software Product Line Practice* and provided valuable feedback on the draft companion. Challenges and solutions within the DoD community were discussed.

The working groups focused on specific practice areas within software engineering, technical management, and organizational management. As in previous workshops, the empirical and anecdotal evidence that the workshop participants brought to the discussion significantly enhanced our current understanding of the practices and issues as they apply to the DoD. Traditional DoD acquisition strategies are not naturally conducive to software product lines, but product line practice is possible within the DoD, and there is a growing number of DoD organizations that are taking a product line approach.

Within the DoD there needs to be increased awareness about DoD product line activities that might be relevant. It is critical for the DoD to think more strategically and to share information and outcomes between different areas. These outcomes could help to prevent duplication and redundant development.

In an effort to expand both the information base and the DoD community interested in software product lines, the SEI was encouraged by the participants to continue to hold similar workshops.

The results of this workshop have been incorporated into the companion, which will continue to be refined and revised as the technology matures and as we continue to receive feedback and to work with the growing community of software engineers championing a product line approach. If you have any comments on this report or are using a product line approach in the development or acquisition of software-intensive systems for the DoD and would like to participate in a future workshop, please send email to Linda Northrop at lmn@sei.cmu.edu.

Appendix Workshop Participant Feedback on the Acquisition Companion

The participants provided some excellent feedback on the draft companion. Because this information is primarily useful to the authors of the companion, it was placed in this appendix.

- The companion gives only a static view of the practice areas. As noted in the framework, there is dynamic interaction among the practices. Can you provide a look at the dynamic behavior of practice areas in the companion?
- The production plan should have more emphasis in the companion. At least it should be discussed in the “Software System Integration” practice area, if not before.
- For the “Operations” practice area, consider discussing deployment issues, such as the phase-in of a new system that replaces an old system.
- The large number of practice areas (29) is daunting. Can you provide managers with a more manageable group to tackle them? A related issue is that the size of the companion may also be overwhelming. We need a “non-scary” introduction for managers.
- There ought to be an easy way to download the companion and the framework so that people can have a local copy on their computer. A Web-based document doesn’t help you when you aren’t hooked up to the Web.
- We need some help knowing which product line metrics are useful to collect.

Bibliography

All URLs are valid as of the publication date of this document.

- [Bass 97]** Bass, L.; Clements, P.; Cohen, S.; Northrop, L.; & Withey, J. *Product Line Practice Workshop Report* (CMU/SEI-97-TR-003, ADA327610). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1997.
<<http://www.sei.cmu.edu/publications/documents/97.reports/97tr003/97tr003abstract.html>>.
- [Bass 98]** Bass, L.; Chastek, G.; Clements, P.; Northrop, L.; Smith, D.; & Withey, J. *Second Product Line Practice Workshop Report* (CMU/SEI-98-TR-015, ADA354691). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1998.
<<http://www.sei.cmu.edu/publications/documents/98.reports/98tr015/98tr015abstract.html>>.
- [Bass 99]** Bass, L.; Campbell, G.; Clements, P.; Northrop, L.; & Smith, D. *Third Product Line Practice Workshop Report* (CMU/SEI-99-TR-003, ADA361391). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1999.
<<http://www.sei.cmu.edu/publications/documents/99.reports/99tr003/99tr003abstract.html>>.
- [Bass 00]** Bass, L.; Clements, P.; Donohoe, P.; McGregor, J.; & Northrop, L. *Fourth Product Line Practice Workshop Report* (CMU/SEI-2000-TR-002, ADA375843). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2000.
<<http://www.sei.cmu.edu/publications/documents/00.reports/00tr002.html>>.
- [Bass 03]** Bass, L.; Clements, P.; & Kazman, R. *Software Architecture in Practice*, 2nd ed. Reading, MA: Addison-Wesley, 2003.

[Bergey 98]

Bergey, J.; Clements, P.; Cohen, S.; Donohoe, P.; Jones, L.; Krut, R.; Northrop, L.; Tilley, S.; Smith, D.; & Withey, J. *DoD Product Line Practice Workshop Report* (CMU/SEI-98-TR-007, ADA346252). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1998.
<<http://www.sei.cmu.edu/publications/documents/98.reports/98tr007/98tr007abstract.html>>.

[Bergey 99a]

Bergey, J.; Campbell, G.; Clements, P.; Cohen, S.; Jones, L.; Krut, R.; Northrop, L.; & Smith, D. *Second DoD Product Line Practice Workshop Report* (CMU/SEI-99-TR-015, ADA375845). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1999. <<http://www.sei.cmu.edu/publications/documents/99.reports/99tr015/99tr015abstract.html>>.

[Bergey 99b]

Bergey, J.; Fisher, M.; Jones, L.; & Kazman, R. *Software Architecture Evaluation with ATAM in the DoD Systems Acquisition Context* (CMU/SEI-99-TN-012, ADA377450). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1999.
<<http://www.sei.cmu.edu/publications/documents/99.reports/99tn012/99tn012abstract.html>>.

[Bergey 00a]

Bergey, J.; Campbell, G.; Cohen, S.; Fisher, M.; Jones, L.; Krut, R.; Northrop, L.; O'Brien, W.; Smith, D.; & Soule, A. *Third DoD Product Line Practice Workshop Report* (CMU/SEI-2000-TR-024, ADA387259). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2000.
<<http://www.sei.cmu.edu/publications/documents/00.reports/00tr024.html>>.

[Bergey 00b]

Bergey, J.; Fisher, M.; Gallagher, B.; Jones, L.; & Northrop, L. *Basic Concepts of Product Line Practice for the DoD* (CMU/SEI-2000-TN-001, ADA375859). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2000.
<<http://www.sei.cmu.edu/publications/documents/00.reports/00tn001.html>>.

- [Bergey 01]** Bergey, J. et al. *Fourth DoD Product Line Practice Workshop Report* (CMU/SEI-2001-TR-017, ADA399205). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2001. <<http://www.sei.cmu.edu/publications/documents/01.reports/01tr017.html>>.
- [Bergey 03]** Bergey, J.; Campbell, G.; Cohen, S.; Fisher, M.; Gallagher, B.; Jones, L.; Northrop, L.; & Soule, A. *Software Product Line Acquisition – A Companion to A Framework for Software Product Line Practice, Version 2.0*. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003. <<http://www.sei.cmu.edu/plp/companion.html>>.
- [Brownsword 96]** Brownsword, L. & Clements, P. *A Case Study in Successful Product Line Development* (CMU/SEI-96-TR-016, ADA315802). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1996. <<http://www.sei.cmu.edu/publications/documents/96.reports/96.tr.016.html>>.
- [Chastek 02]** Chastek, G., ed. *Proceedings of the Second Software Product Lines Conference (SPLC2)* (LNCS 2370). San Diego, CA, August 19-22, 2002. New York, NY: Springer-Verlag, 2002.
- [Clements 01]** Clements, P. et al. *Fifth Product Line Practice Workshop Report* (CMU/SEI-2001-TR-027, ADA272355) Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2001. <<http://www.sei.cmu.edu/publications/documents/01.reports/01tr027.html>>.
- [Clements 02a]** Clements, P. & Northrop, L. *Software Product Lines: Practices and Patterns*. Reading, MA: Addison-Wesley, 2002.
- [Clements 02b]** Clements, P.; & Northrop, L. *A Framework for Software Product Line Practice, Version 4.1*. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2002. <<http://www.sei.cmu.edu/plp/framework.html>>.

[Cohen 02]

Cohen, S.; Dunn, E. & Soule, A. *Successful Product Line Development and Sustainment: A DoD Case Study in Successful Product Line Development* (CMU/SEI-2002-TN-018, ADA407788). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2002. <<http://www.sei.cmu.edu/publications/documents/02.reports/02tn018.html>>.

[Donohoe 00]

Donohoe, P. *Software Product Lines: Experience and Research Directions*. Proceedings of the First Software Product Line Conference (SPLC1), Denver, CO, Aug. 28-31, 2000. Boston, MA: Kluwer Academic Publishers, 2000.

[SPC 93]

Software Productivity Consortium. *Reuse-Driven Software Process Guidebook*, SPC-92019-CMC, Version 02.00.03. Herndon, VA: Software Productivity Consortium Services Corporation, 1993.

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.				
1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE June 2003	3. REPORT TYPE AND DATES COVERED Final		
4. TITLE AND SUBTITLE Fifth DoD Product Line Practice Workshop Report		5. FUNDING NUMBERS F19628-00-C-0003		
6. AUTHOR(S) John Bergey, Sholom Cohen, Matthew Fisher, Lawrence Jones, Linda Northrop, William O'Brien				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213		8. PERFORMING ORGANIZATION REPORT NUMBER CMU/SEI-2003-TR-007		
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116		10. SPONSORING/MONITORING AGENCY REPORT NUMBER ESC-TR-2003-007		
11. SUPPLEMENTARY NOTES				
12A DISTRIBUTION/AVAILABILITY STATEMENT Unclassified/Unlimited, DTIC, NTIS		12B DISTRIBUTION CODE		
13. ABSTRACT (MAXIMUM 200 WORDS) The Software Engineering Institute (SEI SM) held the Fifth Department of Defense (DoD) Product Line Practice Workshop in August 2002 in conjunction with the Second Software Product Line Conference (SPLC2). The workshop was a hands-on meeting to identify industry-wide best practices in software product lines; to share DoD product line practices, experiences, and issues; to discuss ways in which specific product line practices are accomplished within the DoD; and to obtain feedback on the Version 2 pre-release draft of <i>Software Product Line Acquisition: A Companion to a Framework for Software Product Line Practice</i> written by the SEI. This report synthesizes the workshop presentations and discussions.				
14. SUBJECT TERMS DoD product line practice, Product Line Practice Framework, product line workshop, software architecture, software product lines, DoD software acquisition		15. NUMBER OF PAGES 52		
16. PRICE CODE				
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL	